



Creación de Aplicaciones Python

Lección 4: Primeros pasos en el entorno Django

ÍNDICE

Lección 4: Primeros pasos en el entorno Django	1
Presentación y objetivos.....	1
1. ¿ Qué es Django ? ¿ Por qué usarlo?.....	2
2. Creando una aplicación en Django	3
2.1 Instalación librería Django	3
2.2 Creación del proyecto Django	6
2.3 Ejecutar la aplicación.....	11
2.4 Creando resto de archivos necesarios.....	15
2.5 Migración de la aplicación.....	28
3. Puntos clave.....	32

Lección 4: Primeros pasos en el entorno Django

PRESENTACIÓN Y OBJETIVOS

En esta lección aprenderemos los primeros pasos para desarrollar una aplicación de Django empezando una aplicación desde cero, con explicación del entorno y de las partes que consta.



Objetivos

- *Conocer que es un Entorno Virtual.*
- *Empezar una Aplicación Django desde cero*
- *Partes de las que consta la aplicación Django.*

1. ¿ QUÉ ES DJANGO ? ¿ POR QUÉ USARLO?

Django es un Framework web que nos permite agilizar las tareas de la programación. Está escrito en Python y es de código abierto.

Características:

- ✓ Fácil de utilizar gracias al lenguaje Python, tanto para programadores expertos como juniors.
- ✓ Maneja en su arquitectura el modelo MVC (Modelo-Vista-Controlador).
- ✓ Documentación completa
- ✓ API de base de datos robusta, también con un mapeador objeto-relacional.
- ✓ URLs basadas en expresiones regulares.

Funcionamiento:

Django se rige bajo el siguiente proceso:

- 1) El usuario hace una petición, a través de una dirección web (URL).
- 2) Django realiza una consulta para saber qué hacer con la petición.
- 3) Una vez que sabe qué tarea realizar le envía la petición a la vista que corresponde.
- 4) La vista realiza las acciones necesarias en la base de datos.
- 5) Si lo necesita también utiliza la definición de los formularios.
- 6) Para después responder a la petición mostrando la página que contiene esa URL.

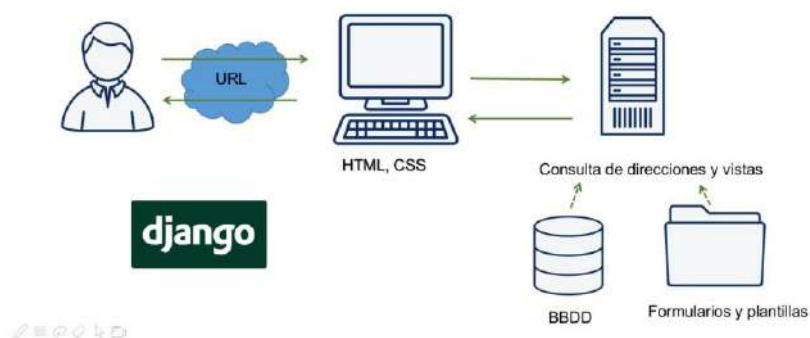


Figura 1.1: Arquitectura de las aplicaciones Django

2. CREANDO UNA APLICACIÓN EN DJANGO

2.1 Instalación librería Django

Creamos el proyecto:

En la carpeta creada con nombre atom:

```
mkdir my_django_web  
  
cd my_django_web/
```

Creamos la carpeta my_django_web donde estará nuestro proyecto:

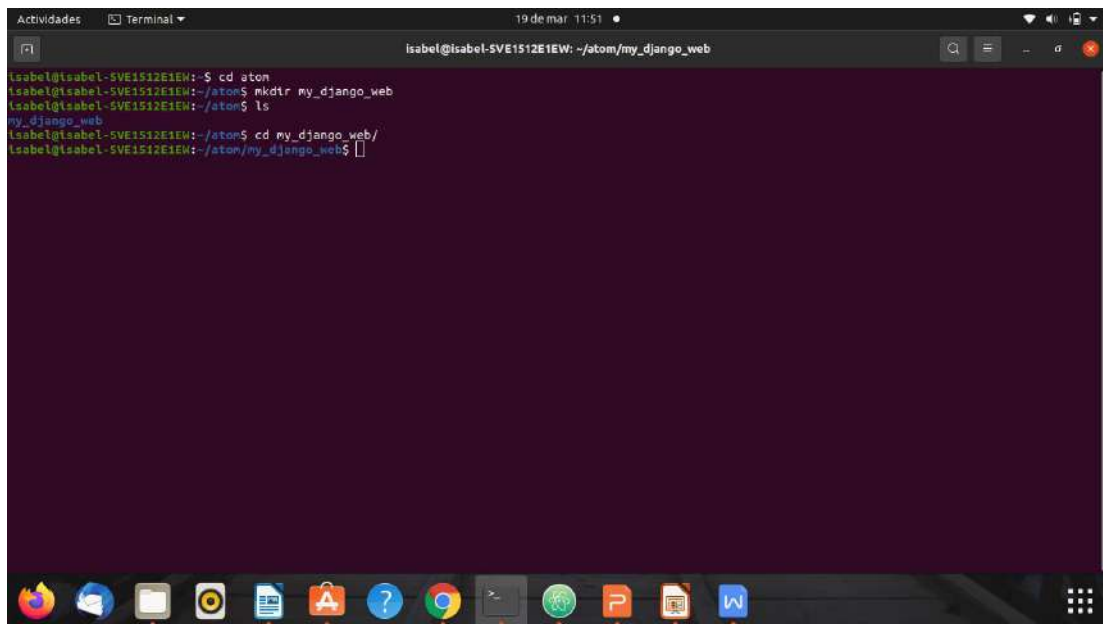


Figura 2.1: Creación de la carpeta del proyecto Django

Creamos el entorno virtual mediante el siguiente comando:

```
virtualenv my_django_app
```

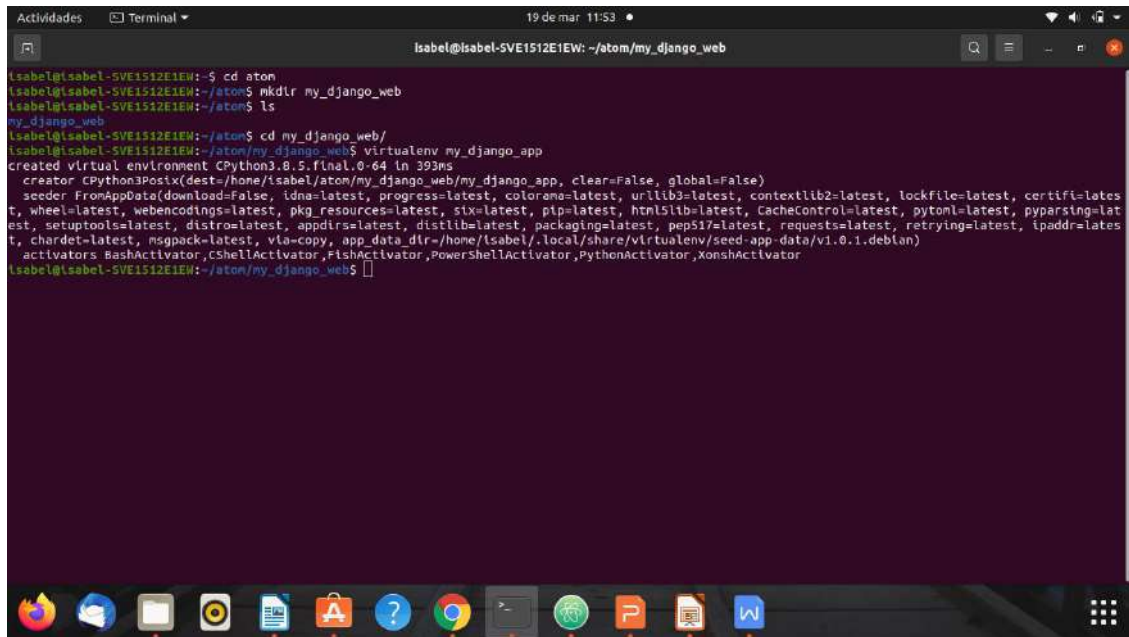


Figura 2.2: Creación del Entorno Virtual

Observamos que en nuestra carpeta creada my_django_web se nos ha creado otra con el Entorno Virtual con el nombre my_django_app

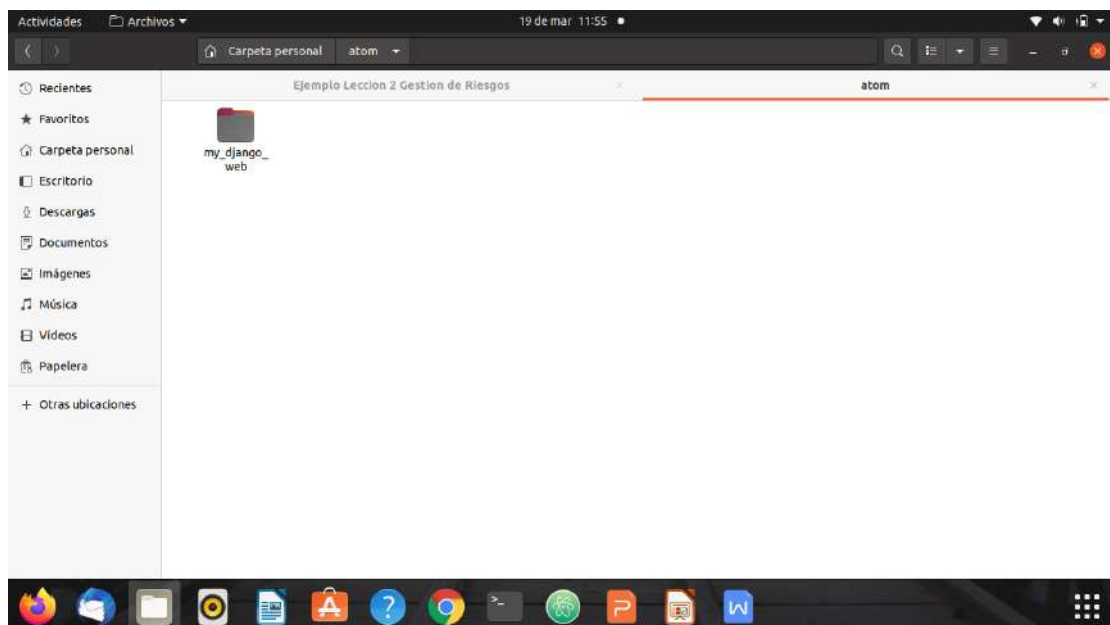
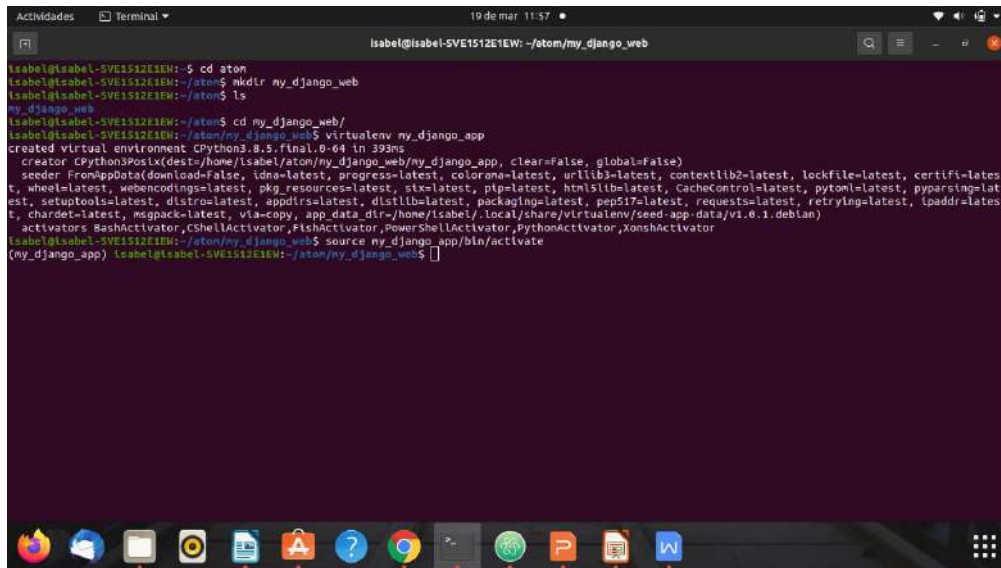


Figura 2.3: Carpeta creada con el Entorno Virtual

Una vez creado accedemos al entorno mediante:

```
source my_django_app/bin/activate
```



```
Isabel@isabel-SVE1512E1EW:~/atom$ cd atom
Isabel@isabel-SVE1512E1EW:~/atom$ mkdir my_django_web
Isabel@isabel-SVE1512E1EW:~/atom$ ls
my_django_web
Isabel@isabel-SVE1512E1EW:~/atom$ cd my_django_web/
Isabel@isabel-SVE1512E1EW:~/atom/my_django_web$ virtualenv my_django_app
created virtual environment CPython3.8.5.final.0-64 in 393ms
creator CPython3Posix(dest=/home/isabel/atom/my_django_web/my_django_app, clear=False, global=False)
seeder FromAppData(download=False, idna=latest, progress=latest, colorama=latest, urllib3=latest, contextlib2=latest, lockfile=latest, certifi=late
t, wheel=latest, webencodings=latest, pkg_resources=latest, six=latest, pip=latest, html5lib=latest, CacheControl=latest, pytoml=latest, pyparsing=late
est, setuptools=latest, distro=latest, appdirs=latest, distlib=latest, packaging=latest, pep517=latest, requests=latest, retrying=latest, ipaddr=lates
t, chardet=latest, msgpack=latest, via-coppy, app data dir=/home/isabel/.local/share/virtualenv/seed-app-data/vi.0.1.debian)
activators BashActivator,CShellActivator,FishActivator,PowerShellActivator,PythonActivator,XonshActivator
Isabel@isabel-SVE1512E1EW:~/atom/my_django_web$ source my_django_app/bin/activate
(my_django_app) Isabel@isabel-SVE1512E1EW:~/atom/my_django_web$
```

Figura 2.4: Activación del Entorno Virtual

Nos aparece entre paréntesis el entorno en el cual estamos trabajando.

A continuación, necesitamos instalar la librería de Django para poder crear nuestras aplicaciones, para ello:

```
pip install Django
```

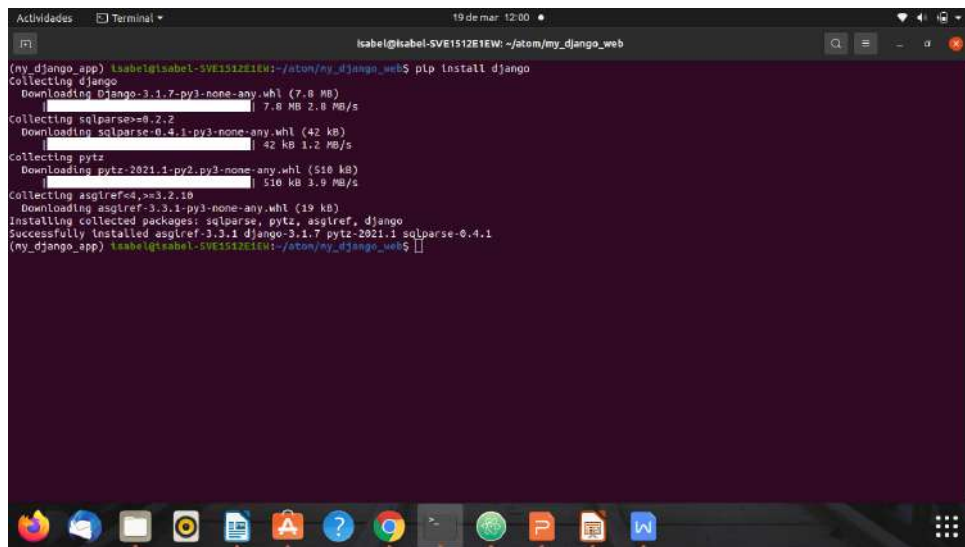


Figura 2.5: Instalación de Django

2.2 Creación del proyecto Django

Ya tenemos instalada la librería para crear nuestro proyecto con Django.

Para crear un nuevo proyecto ponemos:

```
django-admin startproject my_projectWeb
```

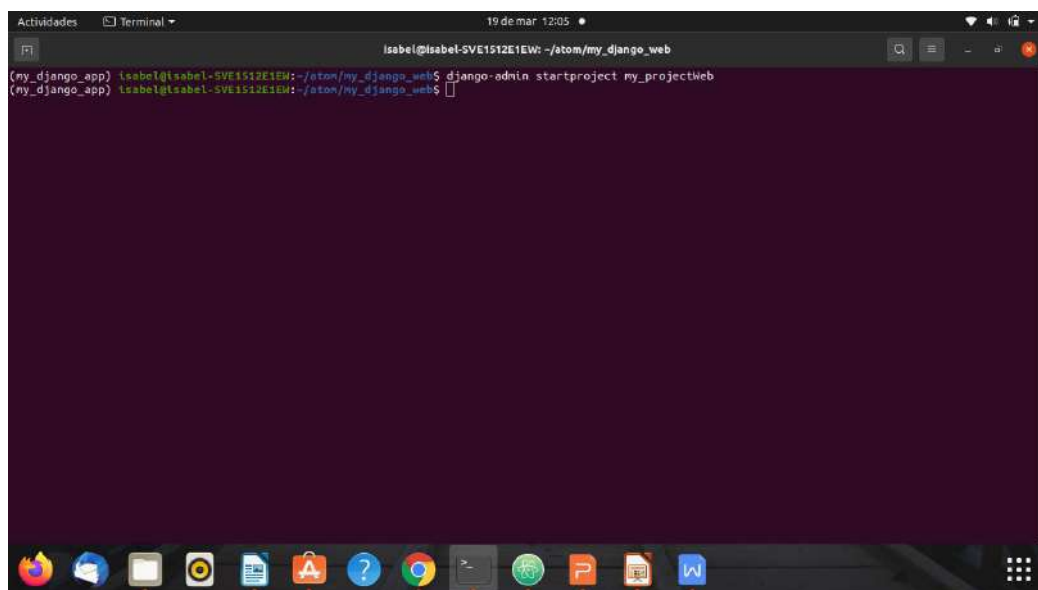


Figura 2.6: Creación del proyecto Django

Si vamos a nuestra ubicación veremos que se nos ha creado una carpeta con ese nombre:

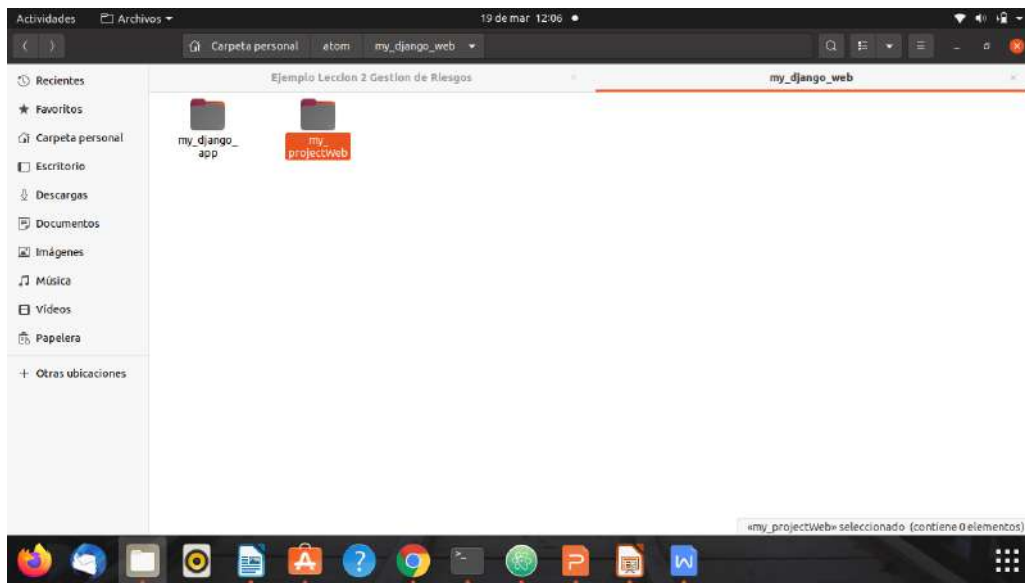


Figura 2.7: Vista del proyecto Django

Ahora tenemos la siguiente estructura:

```

my_django_web/ ←→
|-- my_projectWeb ←→
Proyecto django
|   |-- my_projectWeb/
|   |   |-- __init__.py
|   |   |-- settings.py
|   |   |-- urls.py
|   |   |-- wsgi.py
|   |-- manage.py
+-- my_django_app/ ←→
  
```

carpeta de más alto nivel
carpeta del

carpeta del entorno virtual

manage.py: Un atajo para usar la utilidad de línea de comando django-admin. Es usado para ejecutar comandos de administración relacionados con nuestro proyecto. Lo usaremos para ejecutar el desarrollo en servidores, pruebas, migraciones y muchos más.

init.py: Este archivo vacío le dice a Python que esta carpeta es un paquete.

settings.py: Este archivo contiene la configuración de todo el proyecto. ¡Lo estaremos usando todo el tiempo!

urls.py: Este archivo es el responsable de mapear las rutas y caminos (paths) en nuestro proyecto. Por ejemplo, si quieres mostrar algo en la URL /about/, tienes que mapearlo aquí primero.

wsgi.py: Este archivo es simplemente una interfaz de puerta de enlace usada para despliegues.

Ahora realizaremos una aplicación donde desarrollaremos nuestra aplicación de web basada en django para ello vamos a la carpeta de proyecto:

```
cd my_projectWeb
```

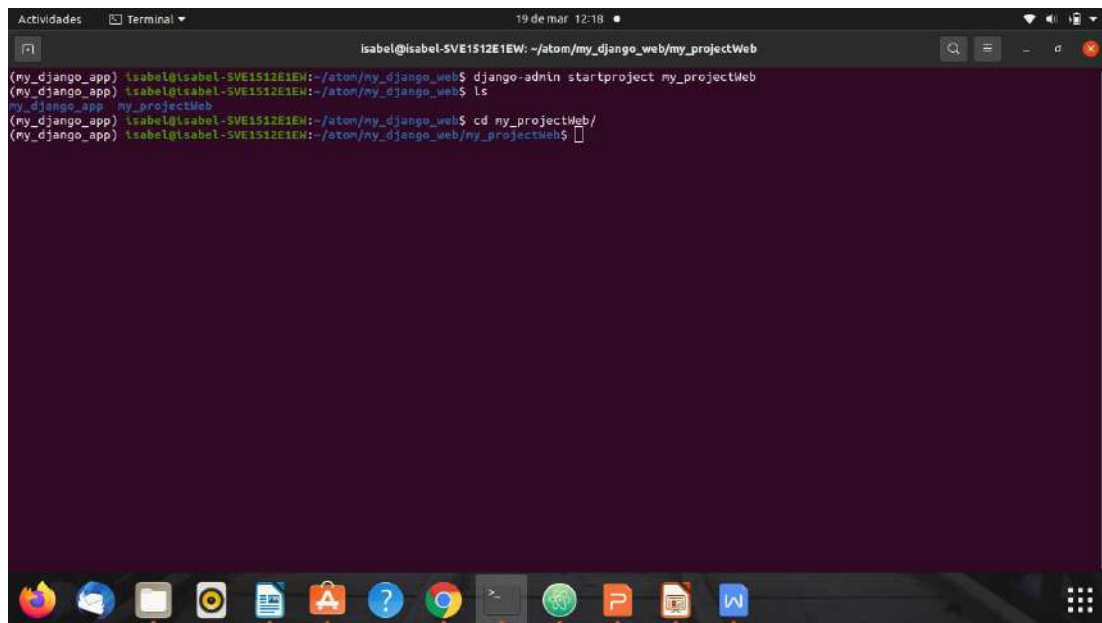


Figura 2.8: Ejemplo de navegación en carpetas del proyecto Django

Para crear la aplicación Django ejecutamos:

```
django-admin startapp mytApp
```

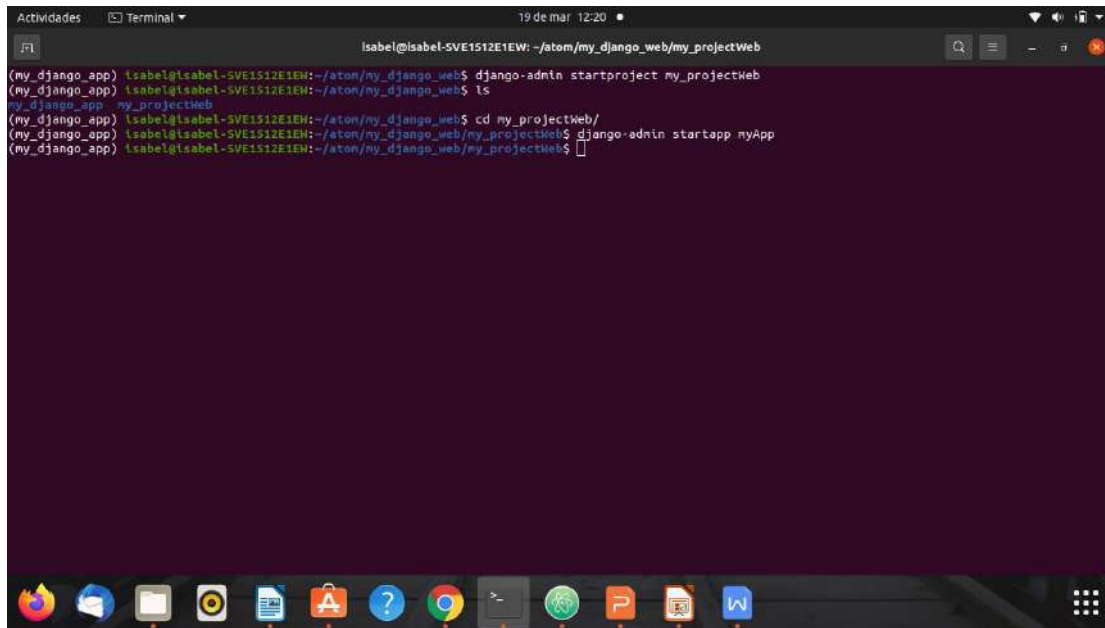


Figura 2.9: Creación de la aplicación Django

Vamos a la carpeta de my_proyectWeb y vemos que se nos ha creado otra carpeta con el nombre myApp:

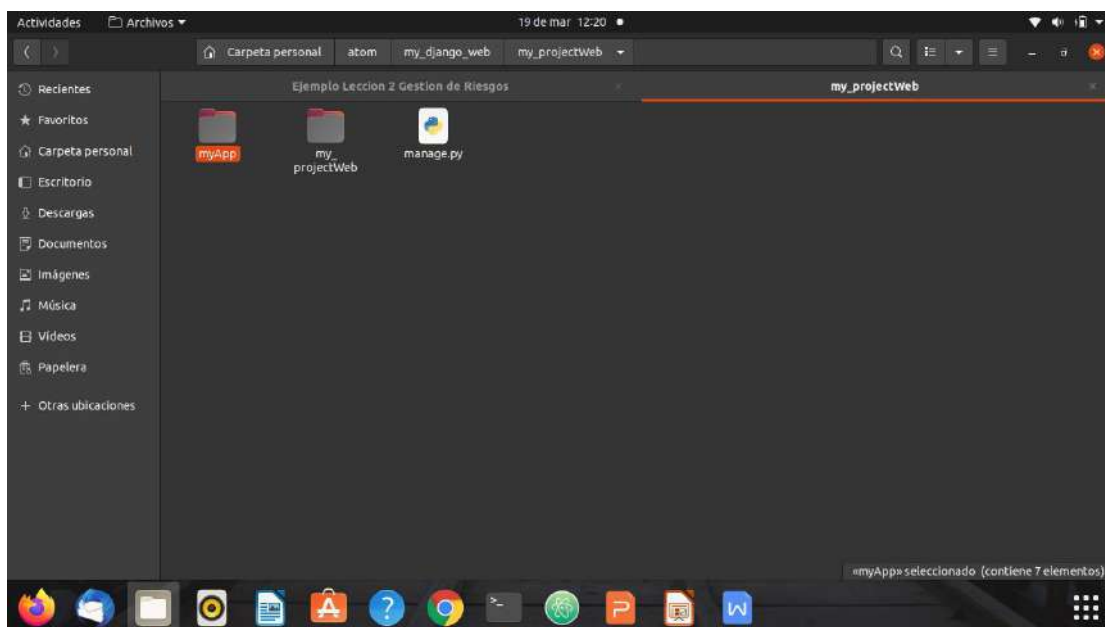


Figura 2.10: Vista de la aplicación Django creada

Ahora el proyecto tiene la siguiente estructura:

```

my_django_web/
|-- my_projectWeb/
|   |-- myApp/ → nuestra nueva
|   |   aplicación Django!
|   |   |-- migrations/
|   |   |   |-- __init__.py
|   |   |-- __init__.py
|   |   |-- admin.py
|   |   |-- apps.py
|   |   |-- models.py
|   |   |-- tests.py
|   |   |-- views.py
|   |-- my_projectWeb/
|   |   |-- __init__.py
|   |   |-- settings.py
|   |   |-- urls.py
|   |   |-- wsgi.py
|   |-- manage.py
+-- my_django_app/

```

Aclaraciones:

migrations: aquí Django almacena algunos archivos para mantener el registro de los cambios que creas en el archivo `models.py`, para manteniéndolos sincronizados.

admin.py: este es un archivo de configuración para una aplicación compilada de Django llamada Django Admin.

apps.py: este es un archivo de configuración de la aplicación en cuestión.

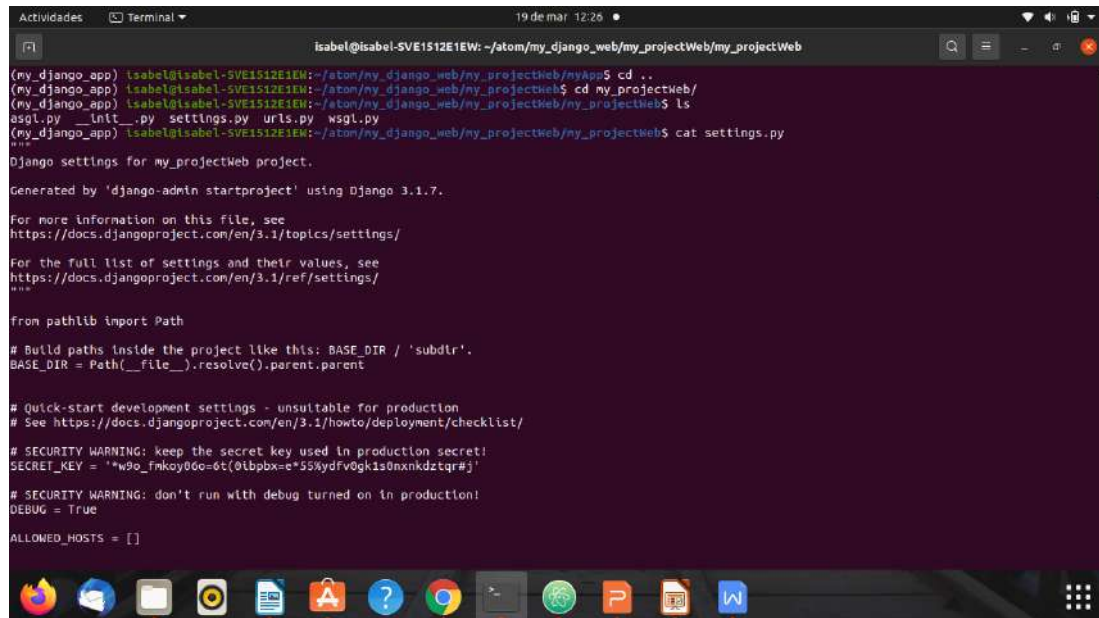
models.py: aquí es donde definimos las entidades de nuestra aplicación web. Los modelos son traducidos automáticamente por Django a tablas de base de datos.

tests.py: este archivo es utilizado para escribir pruebas unitarias para la aplicación.

views.py: este es un archivo donde manejamos el ciclo de solicitudes/respuestas de nuestra aplicación web. Ahora que hemos creado nuestra primera aplicación, vamos a configurarla para usarla.

2.3 Ejecutar la aplicación

Para ejecutar nuestra aplicación es necesario ir a settings.py e intenta encontrar la variable INSTALLED_APPS:



```

Actividades Terminal 19 de mar 12:26
isabel@isabel-SVE1512E1EW: ~/atom/my_django_web/my_projectWeb/my_projectWeb

(my_django_app) isabel@isabel-SVE1512E1EW:~/atom/my_django_web/my_projectWeb/my_app$ cd ..
(my_django_app) isabel@isabel-SVE1512E1EW:~/atom/my_django_web/my_projectWeb$ cd my_projectWeb/
(my_django_app) isabel@isabel-SVE1512E1EW:~/atom/my_django_web/my_projectWeb/my_projectWeb$ ls
osgl.py __init__.py settings.py urls.py wsgi.py
(my_django_app) isabel@isabel-SVE1512E1EW:~/atom/my_django_web/my_projectWeb/my_projectWeb$ cat settings.py
"""
Django settings for my_projectWeb project.

Generated by 'django-admin startproject' using Django 3.1.7.

For more information on this file, see
https://docs.djangoproject.com/en/3.1/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/3.1/ref/settings/
"""

from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'w9o_rmkoy0oo=ot(0lbpbx=e*55kydFv0gk1s0nxkdtqr#j'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []

```

Figura 2.11: Impresión del archivo settings.py

Settings.py

```

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]

```

Ponemos nuestra aplicación:

```

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    'myApp', ]

```

```
sudo nano settings.py
```

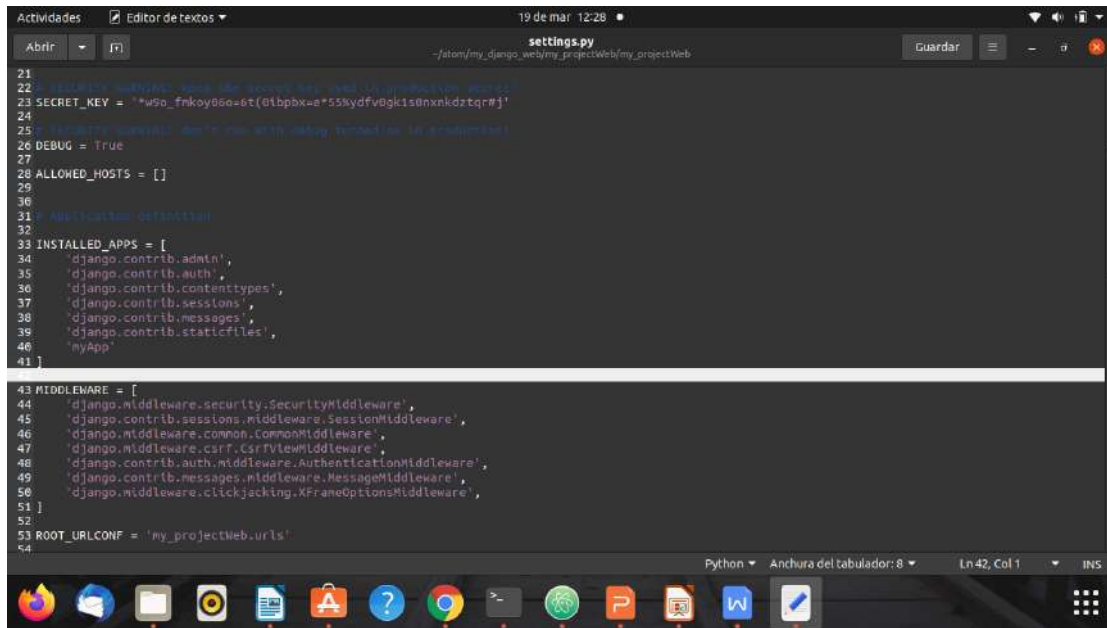


Figura 2.12: Vista del archivo settings.py

Una vez hecho vamos a ver si nuestro proyecto ya puede correr, para ello nos colocamos en la carpeta de `my_proyectWeb` donde tenemos el archivo `manage.py` y ejecutamos:

```
python manage.py runserver
```

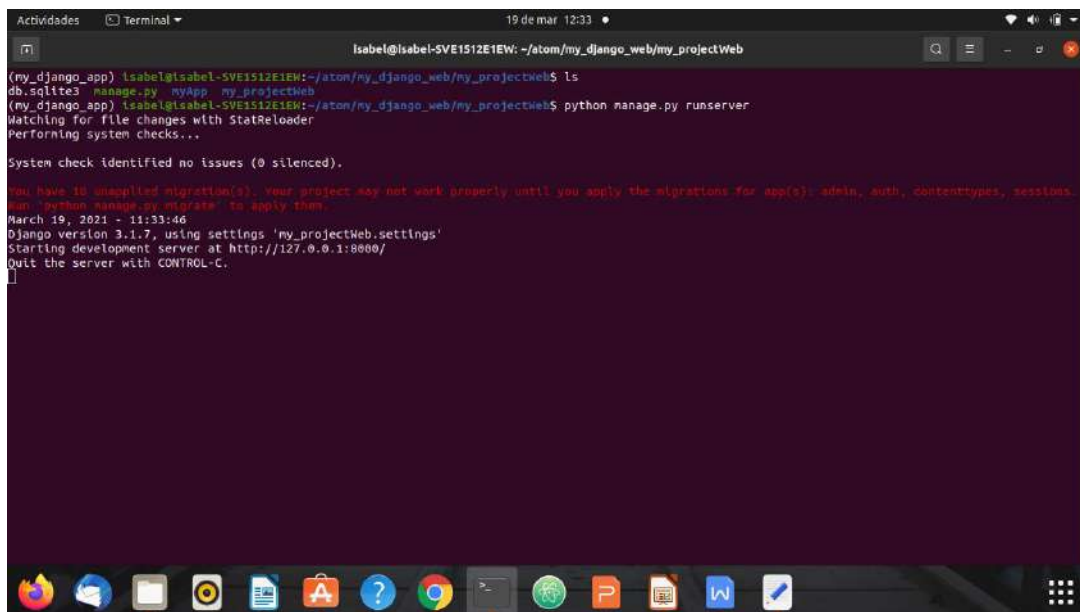


Figura 2.13: Vista de como ejecutar el proyecto Django

Vamos al navegador y ponemos:

`http://localhost:8000/` o `http://127.0.0.1:8000/`

Nos aparece la siguiente pantalla, ya tenemos nuestro primer proyecto Django iniciado:

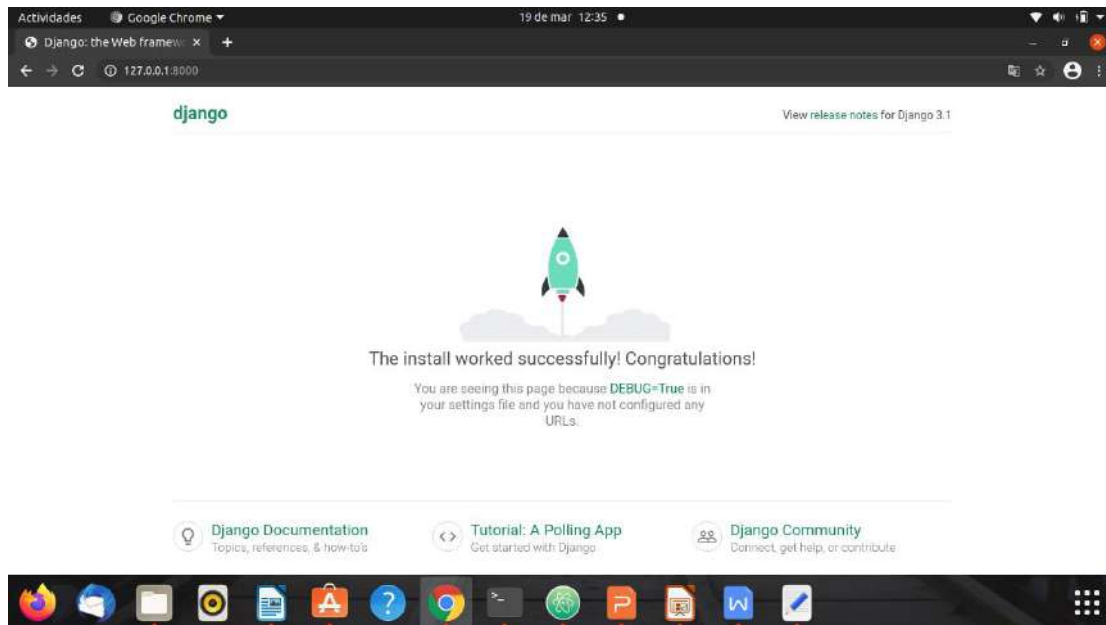


Figura 2.14: Vista de primera ejecución del proyecto Django

Para importar el proyecto a atom lo que debemos hacer es ir a File --> Add Project Folder:

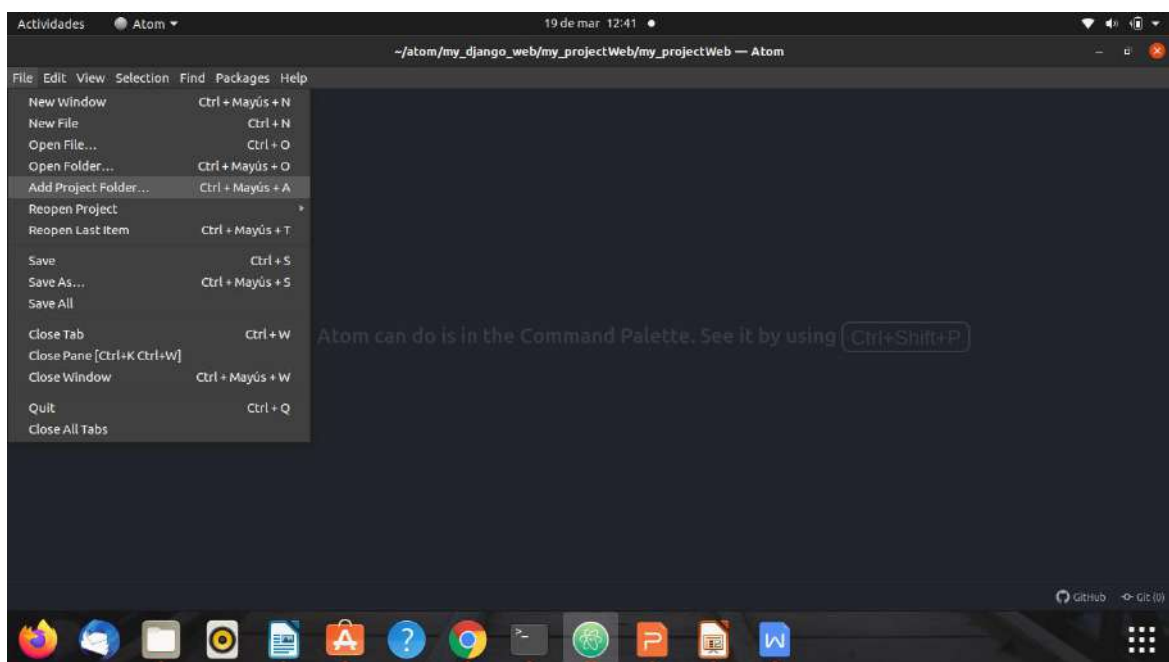


Figura 2.15: Importar el proyecto Django en Atom

Nos abre una ventana donde escogeremos la ruta donde tenemos los proyectos de atom y escogemos my_django_web, aceptar:

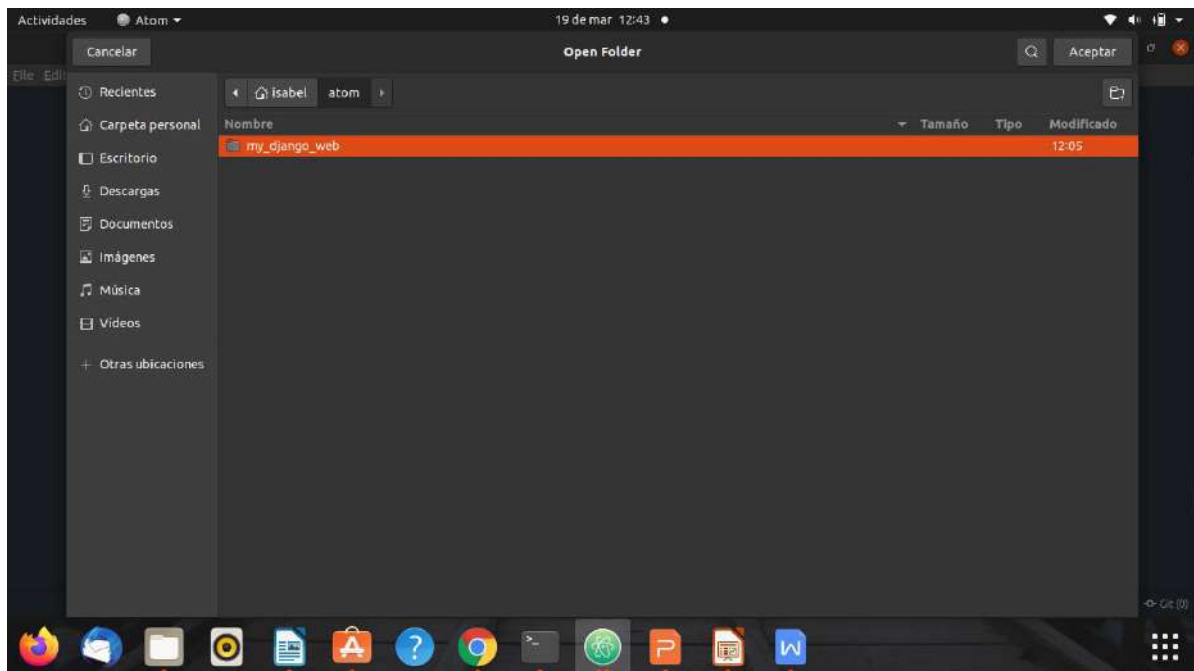


Figura 2.16: Importar el proyecto Django en Atom

Una vez ya importado nos aparecerá nuestro proyecto en la ventana:

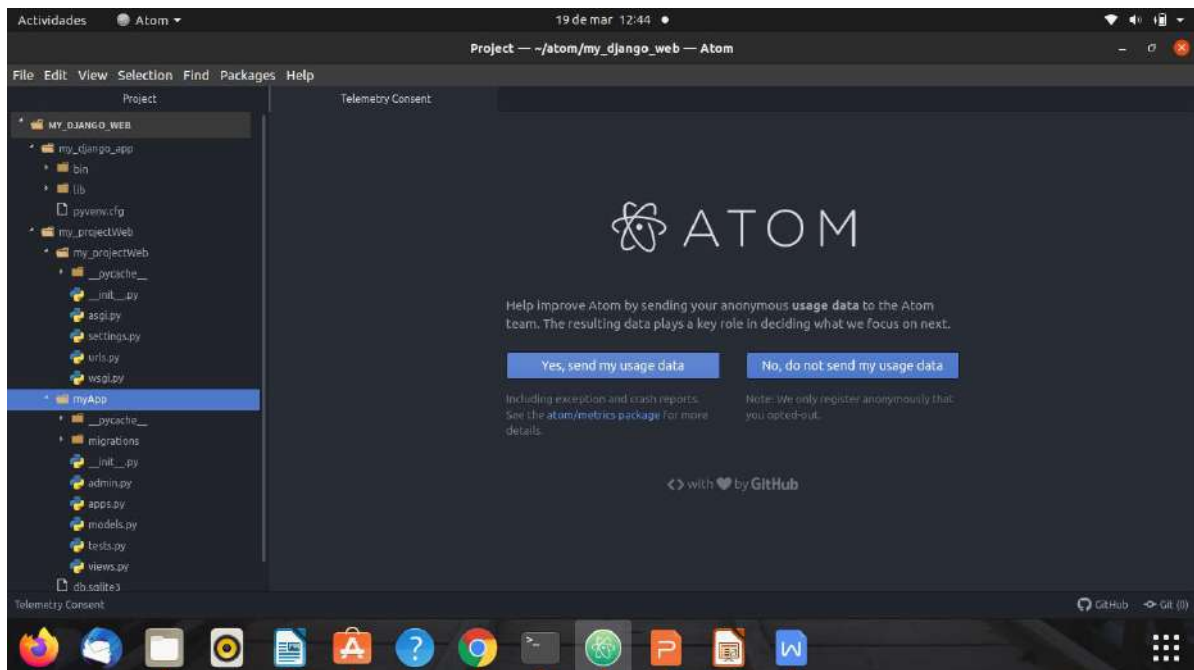
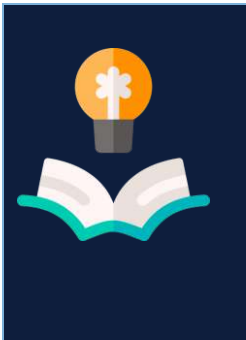


Figura 2.17: Importar el proyecto Django en Atom



Recuerda

Deberás añadir al archivo `settings.py` el nombre de la aplicación que acabamos de crear en `INSTALLED_APPS`.

2.4 Creando resto de archivos necesarios

El archivo `README.md` es un archivo en el cual te encontrar los pasos para ejecutar la aplicación y toda la información relacionada con ella. Para ello nos colocamos en la carpeta de nuestro proyecto y hacemos click con el botón derecho --> New File:

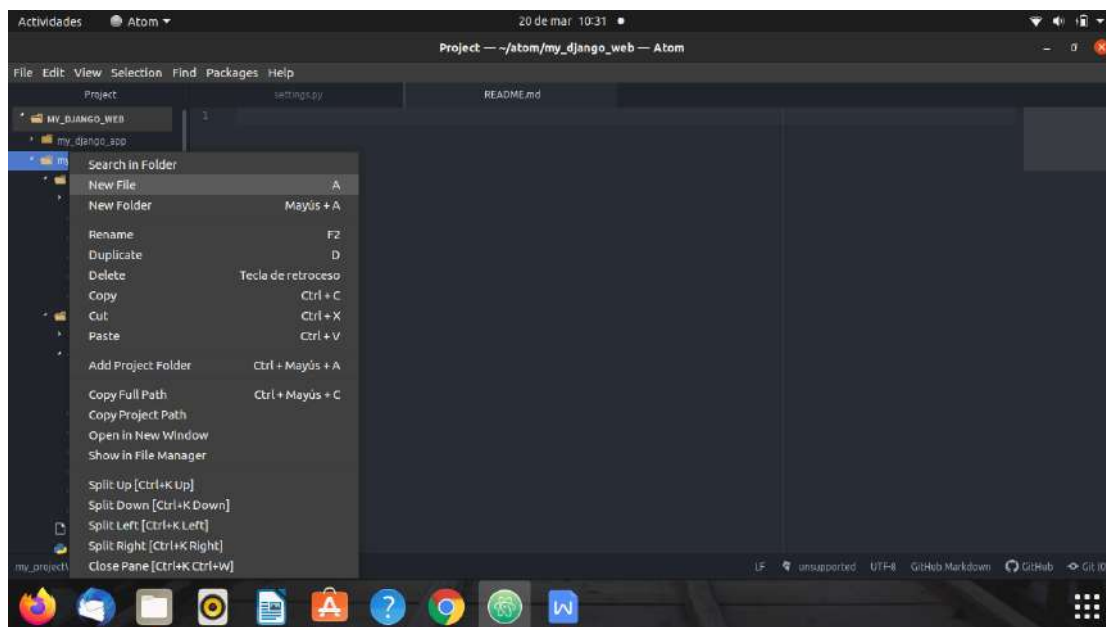


Figura 2.18: Crear una carpeta en Atom

Ponemos el Nombre del archivo:

`README.md`

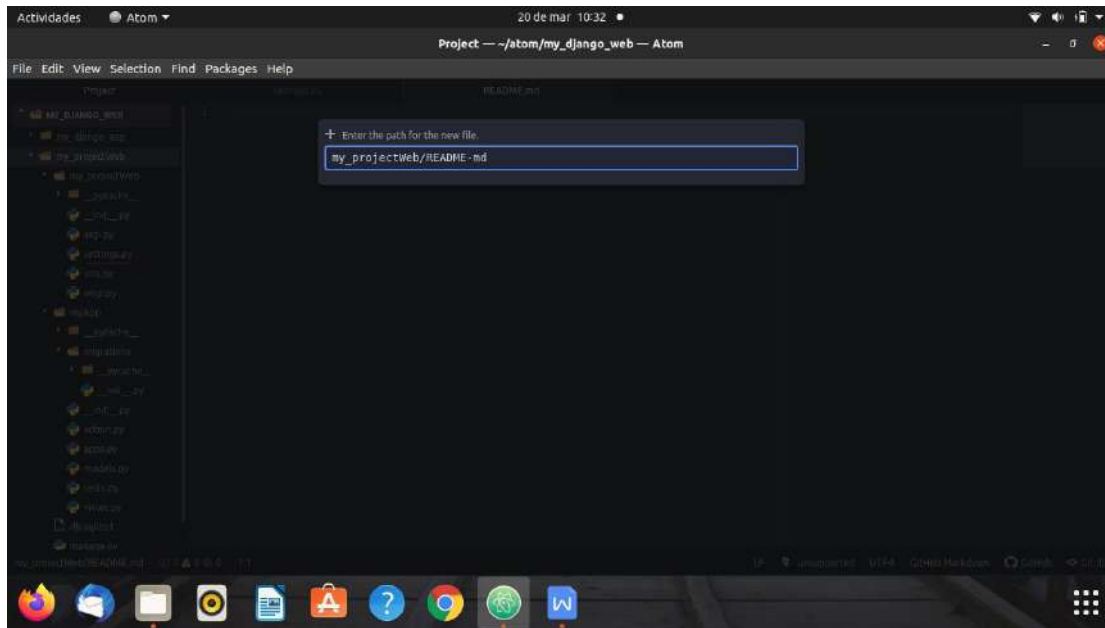


Figura 2.19: Crear una carpeta en Atom

Primeros pasos:

Aquí pondremos las instrucciones para poder ejecutar nuestra aplicación

- 1) Crear el entorno virtual: `virtualenv my_django_app`
- 2) Entrar en el entorno virtual: `source my_django_app/bin/activate`
- 3) Cerrar el entorno virtual: `deactivate`

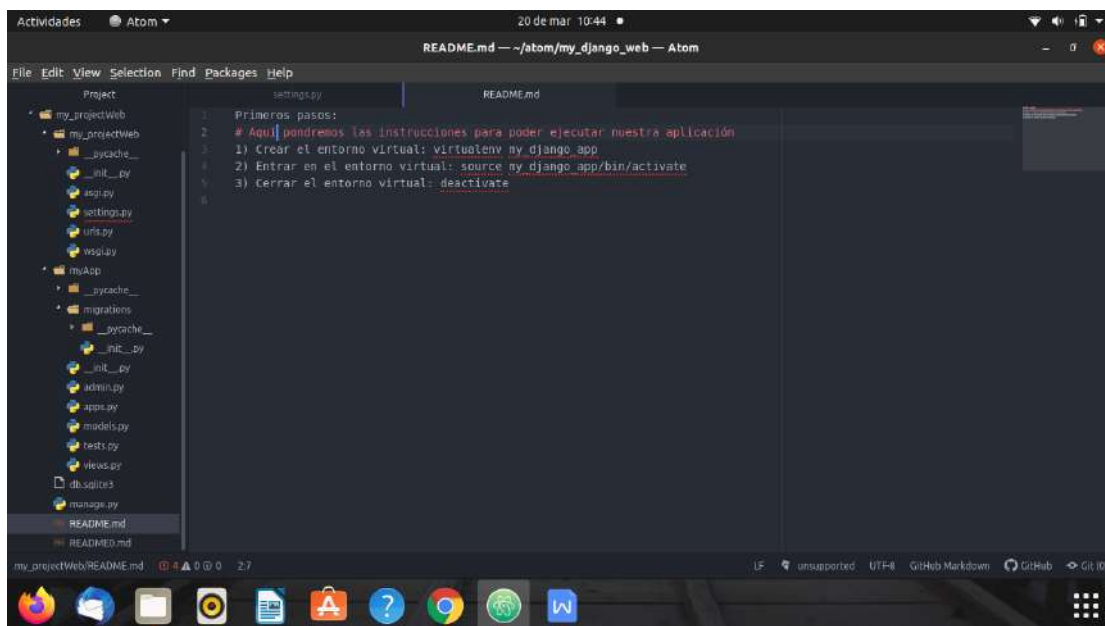
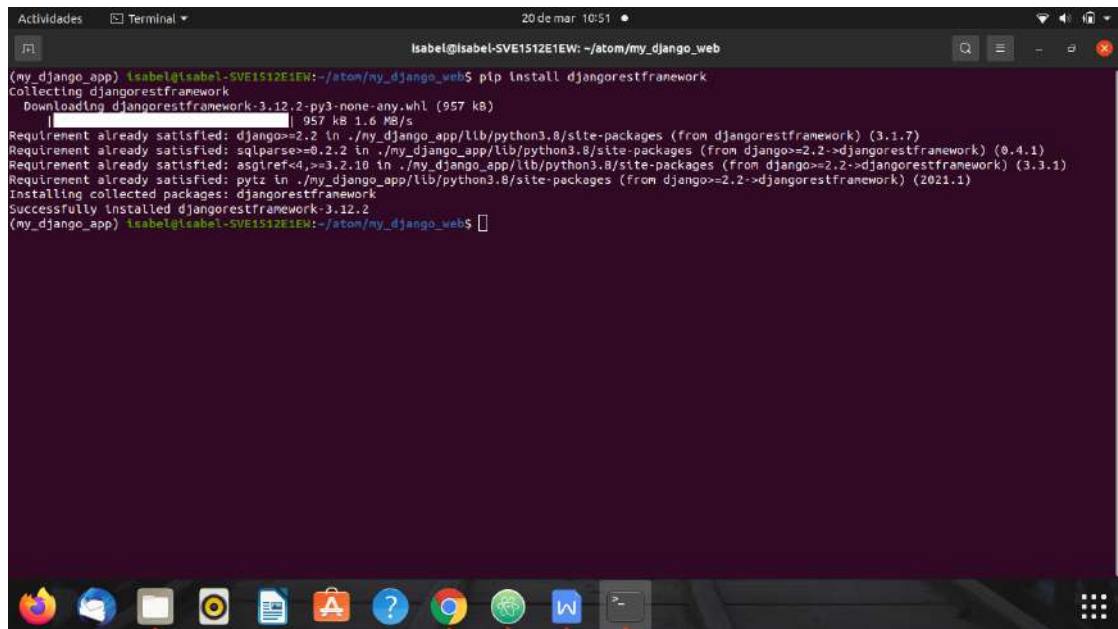


Figura 2.20: Crear archivo README.md

Necesitamos instalar las librerías que vamos a usar para crear nuestro proyecto:

Será necesario para crear la Api Rest la librería django-rest-framework más adelante en el curso veremos para que se va usar:

```
pip install djangorestframework
```



```
Actividades Terminal 20 de mar 10:51
Isabel@Isabel-SVE1512E1EW: ~/atom/my_django_web
(my_django_app) Isabel@Isabel-SVE1512E1EW:~/atom/my_django_web$ pip install djangorestframework
Collecting djangorestframework
  Downloading djangorestframework-3.12.2-py3-none-any.whl (957 kB)
    957 kB 1.6 MB/s
Requirement already satisfied: django>=2.2 in ./my_django_app/lib/python3.8/site-packages (from djangorestframework) (3.1.7)
Requirement already satisfied: sqlparse>=0.2.2 in ./my_django_app/lib/python3.8/site-packages (from django>=2.2->djangorestframework) (0.4.1)
Requirement already satisfied: asgiref<4,>=3.2.10 in ./my_django_app/lib/python3.8/site-packages (from django>=2.2->djangorestframework) (3.3.1)
Requirement already satisfied: pytz in ./my_django_app/lib/python3.8/site-packages (from django>=2.2->djangorestframework) (2021.1)
Installing collected packages: djangorestframework
Successfully installed djangorestframework-3.12.2
(my_django_app) Isabel@Isabel-SVE1512E1EW:~/atom/my_django_web$
```

Figura 2.21: Ejemplo de cómo instalar django rest framework

Podemos ver las librerías que tenemos actualmente instaladas y en que versión se encuentran usando:

```
Pip list
```

```

Actividades Terminal 20 de mar 10:54
Isabel@isabel-SVE1512E1EW: ~/atom/my_django_web
(my_django_app) Isabel@isabel-SVE1512E1EW:~/atom/my_django_web$ pip list
Package Version
-----
appdirs 1.4.3
asgiref 3.3.1
CacheControl 0.12.6
certifi 2019.11.28
chardet 3.0.4
colorama 0.4.3
contextlib2 0.6.0
distlib 0.3.0
distro 1.4.0
Django 3.1.7
django-rest-framework 3.12.2
html5lib 1.0.1
idna 2.8
ipaddr 2.2.0
lockfile 0.12.2
msgpack 0.6.2
packaging 20.3
pep517 0.8.2
pip 20.0.2
pkg-resources 0.0.0
progress 1.5
pyparsing 2.4.6
pytoml 0.1.21
pytz 2021.1
requests 2.22.0
retrying 1.3.3
setuptools 44.0.0
six 1.14.0
sqlparse 0.4.1
urllib3 1.25.8
webencodings 0.5.1

```

Figura 2.22: Ejemplo para ver las librerías y versiones.

Para que esta información sea usada por cualquier otro desarrollador, creamos el archivo con los requerimientos. Vamos a la carpeta del proyecto:

```
cd my_projectWeb/
```

Una vez dentro ejecutamos:

```
pip freeze > requirements.txt
```

Observamos que en nuestro proyecto se ha creado un archivo requirements.txt con nuestras librerías y su versión:

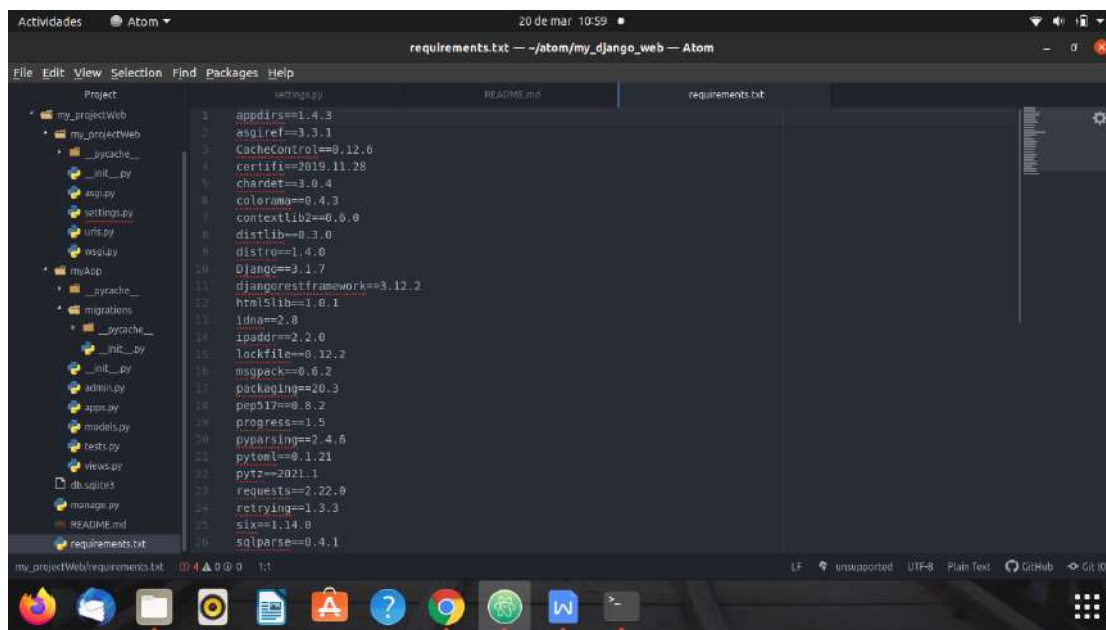


Figura 2.23: Ejemplo de archivo requirements.txt

Añadimos de igual modo al archivo README.md: los siguientes pasos:

- 5) Crear un archivo con los requirements: `pip freeze > requirements.txt`
- 6) Correr la aplicación: `python manage.py runserver`

Quedando de este modo:

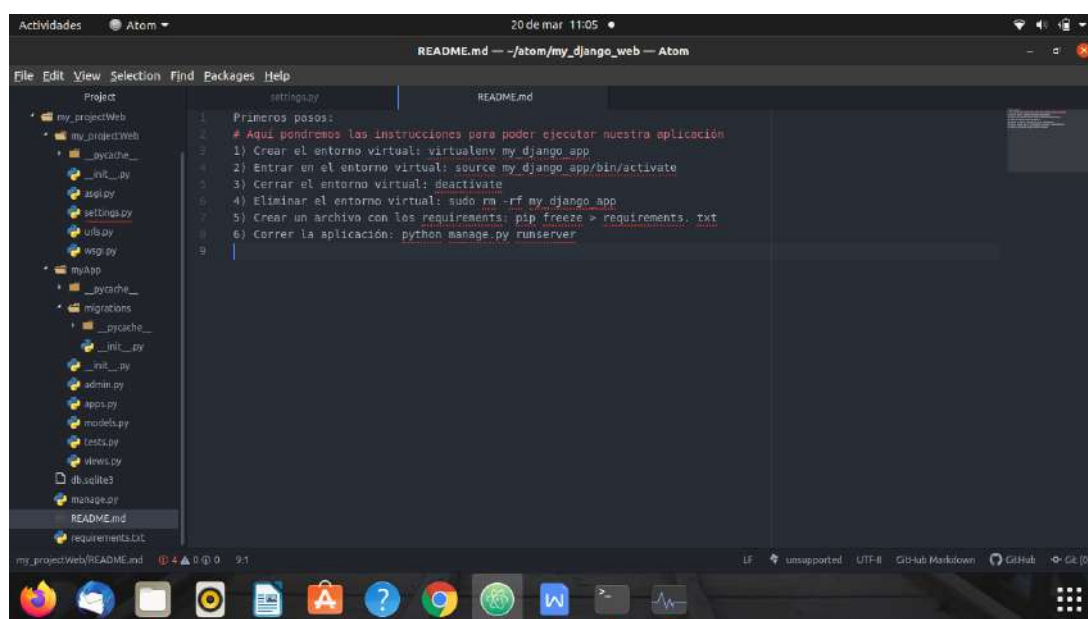


Figura 2.24: Ejemplo de archivo README.md

Creamos las carpetas que vamos a usar en nuestra aplicación:

Vamos a my_projectWeb botón derecho--> New Folder:

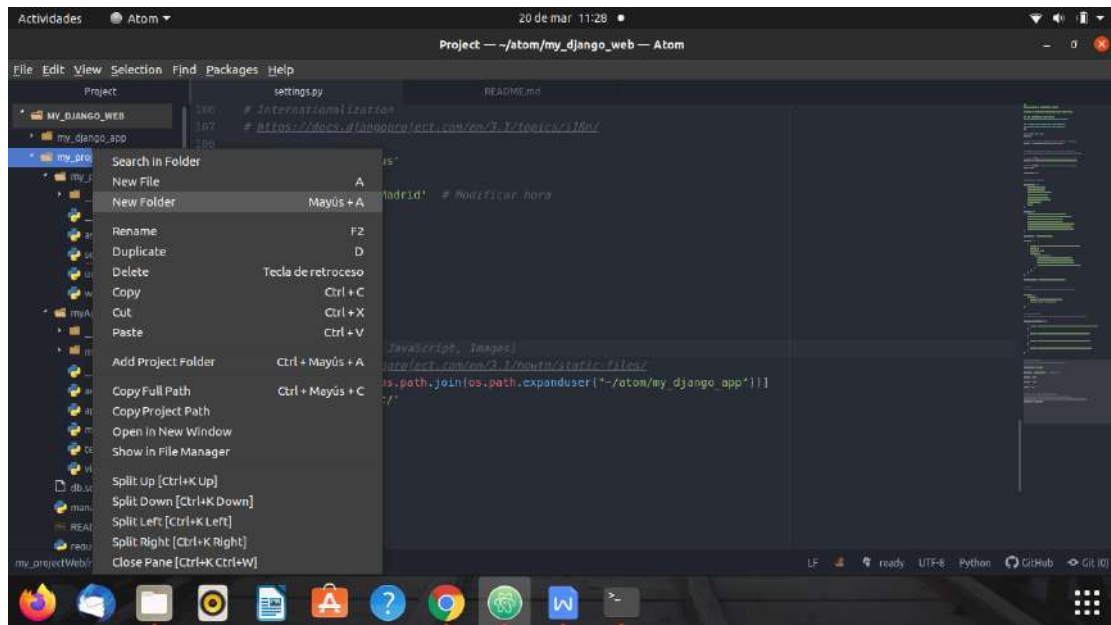


Figura 2.25: Ejemplo de crear nueva carpeta

Pondremos el nombre de nuestra carpeta “templates”:

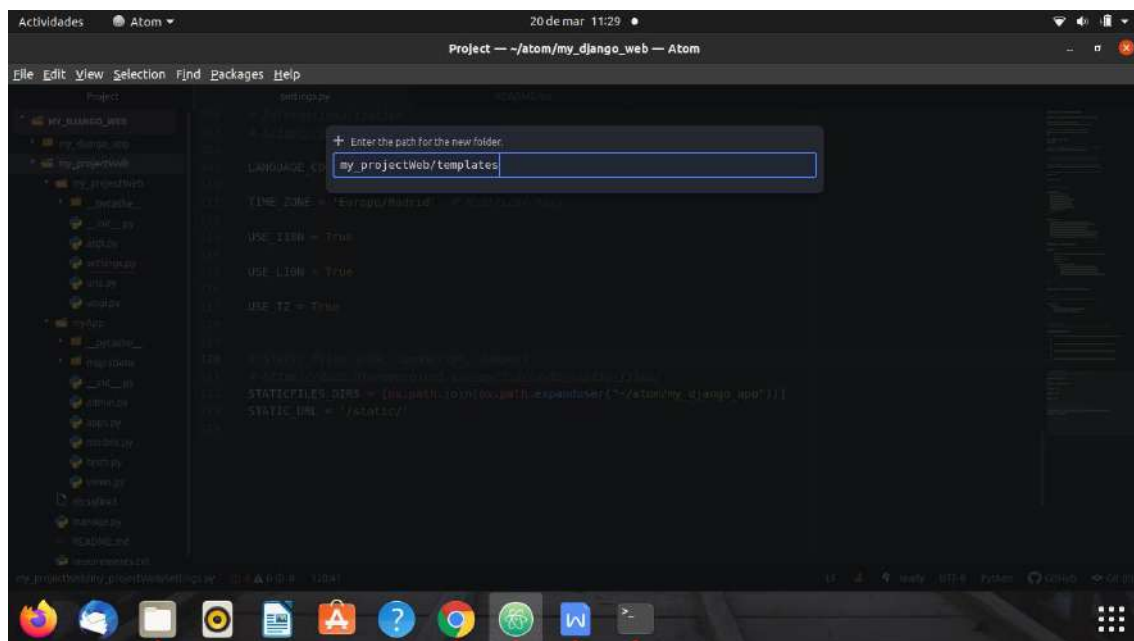


Figura 2.26: Ejemplo de crear la carpeta templates, así se crearán el resto de carpetas

Así haremos con todas ellas.

- ✓ **templates** es la carpeta donde se encuentran las plantillas para html.
- ✓ **static** es la carpeta donde encontraremos las imágenes de nuestro proyecto y los css para el diseño de nuestra página. Carpetas: css--> styles.css y images
- ✓ **media** es la carpeta que contiene los archivos que necesitamos insertar en nuestra aplicación como pdfs, words, txt, csv, etc.
- ✓ **locale** es la carpeta donde se encuentra los archivos para traducir a varios idiomas nuestra aplicación.

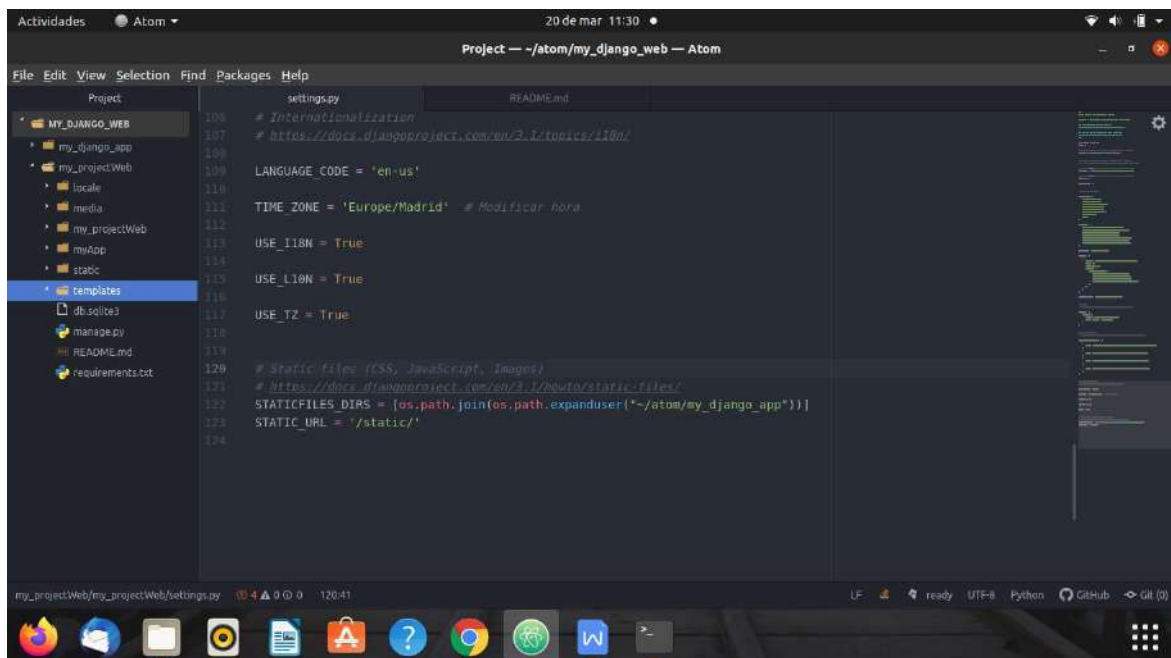


Figura 2.27: Ejemplo de crear la carpeta templates

Settings.py:

Antes de continuar vamos a detenernos en el archivo settings.py:

Build paths inside the project like this: BASE_DIR / 'subdir'.

BASE_DIR = Path(__file__).resolve().parent.parent --> directorio donde se encuentra nuestro proyecto


```
# SECURITY WARNING: keep the secret key used in production secret!

SECRET_KEY = '*w9o_fmky06o=6t0ibpbx=e*55%ydfv0gk1s0nxnkdztqr#j' --
> nuestra llave secreta única generada por cada proyecto

# SECURITY WARNING: don't run with debug turned on in production!

DEBUG = True --> si queremos activar que nos detecte errores, esto en
producción suele estar a FALSE

ALLOWED_HOSTS = [] --> el host donde irá alojada nuestra aplicación suele
ponerse "*" para permitir cualquiera, esto se emplea en producción

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    # Añadidos:
    'myApp',
]

MIDDLEWARE = [ --> necesario para describir los controles en la aplicación
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```


ROOT_URLCONF = 'my_projectWeb.urls' --> ruta de las urls principales

TEMPLATES = [--> ruta dónde se encuentra nuestras plantillas html

```
{
    'BACKEND': 'django.template.backends.django.DjangoTemplates',
    'DIRS': [ ],
    'APP_DIRS': True,
    'OPTIONS': {
        'context_processors': [
            'django.template.context_processors.debug',
            'django.template.context_processors.request',
            'django.contrib.auth.context_processors.auth',
            'django.contrib.messages.context_processors.messages',
        ],
    },
},
]
```

WSGI_APPLICATION = 'my_projectWeb.wsgi.application' --> esto se emplea en producción para dar la ruta del archivo wsgi.py

Database

<https://docs.djangoproject.com/en/3.1/ref/settings/#databases>

DATABASES = { --> conexión a la base de datos esto se hace genera por defecto al crear la aplicación si necesitamos otra conexión a otra base de datos, la describimos en este punto

```
'default': {
    'ENGINE': 'django.db.backends.sqlite3',
    'NAME': BASE_DIR / 'db.sqlite3',
}
}
```

Password validation

<https://docs.djangoproject.com/en/3.1/ref/settings/#auth-password-validators>

```
AUTH_PASSWORD_VALIDATORS = [ --> características de las password
{'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator'},
    {'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator'},
    {'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator'},
    {
                                                                    'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator'},
]
```

Internationalization

<https://docs.djangoproject.com/en/3.1/topics/i18n/>

LANGUAGE_CODE = 'en-us' --> lenguaje principal de la aplicación

TIME_ZONE = 'Europe/Madrid' # Modificar hora --> zona horaria donde nos encontramos

USE_I18N = True --> permitido cambio de idiomas

USE_L10N = True

USE_TZ = True

A continuación, añadiremos al archivo settings.py las rutas de las carpetas necesarias para trabajar con un buen entorno Django:

my_projectWeb--> my_projectWeb --> settings.py

Añadimos:

```
import os
```

Static files (CSS, JavaScript, Images)

<https://docs.djangoproject.com/en/3.1/howto/static-files/>

```

STATICFILES_DIRS =
[os.path.join(os.path.expanduser("~/atom/my_django_web/my_projectWeb"), "static")]

STATIC_URL = '/static/'

# Absolute filesystem path to the directory that will hold user-uploaded files.
# Example: "/home/username/folderProjects/projectname/media"

MEDIA_ROOT =
os.path.join(os.path.expanduser("~/atom/my_django_web/my_projectWeb"), "media")

# URL that handles the media served from MEDIA_ROOT. Make sure to use
# a trailing slash.
# Example: "http://example.com/media/"

MEDIA_URL = "/media/"

# Locale files translate language:

LOCALE_PATHS =
(os.path.join(os.path.expanduser("~/atom/my_django_web/my_projectWeb"), "locale"), )

```

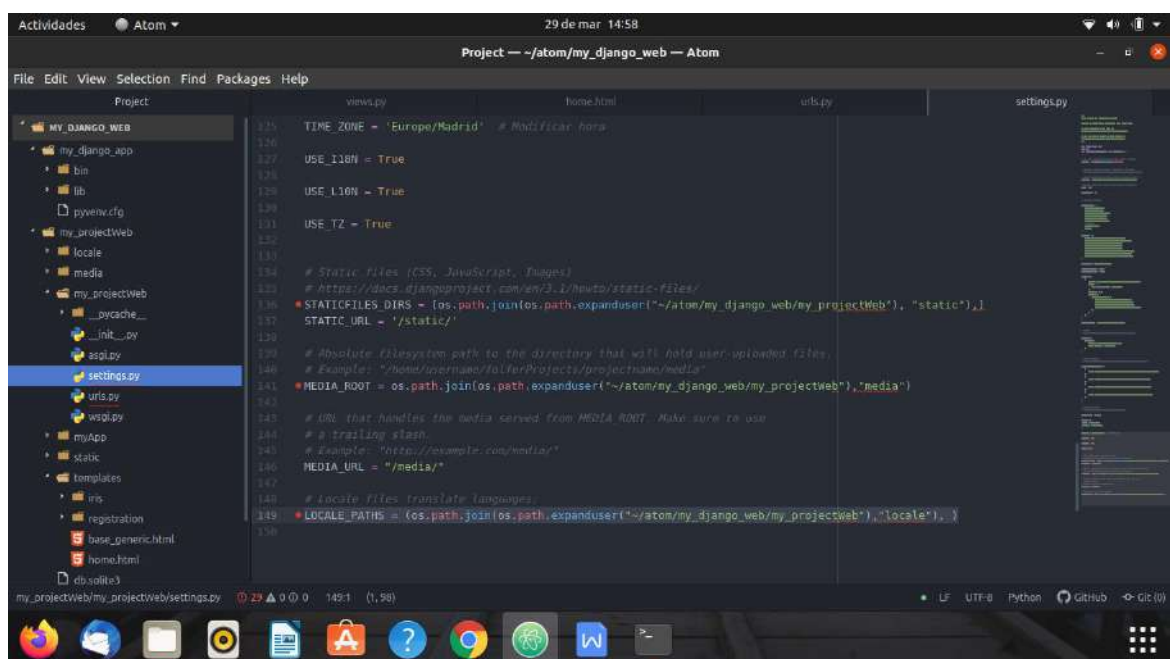


Figura 2.28: Ejemplo de archivo settings.py

Añadimos un archivo a myApp --> urls.py

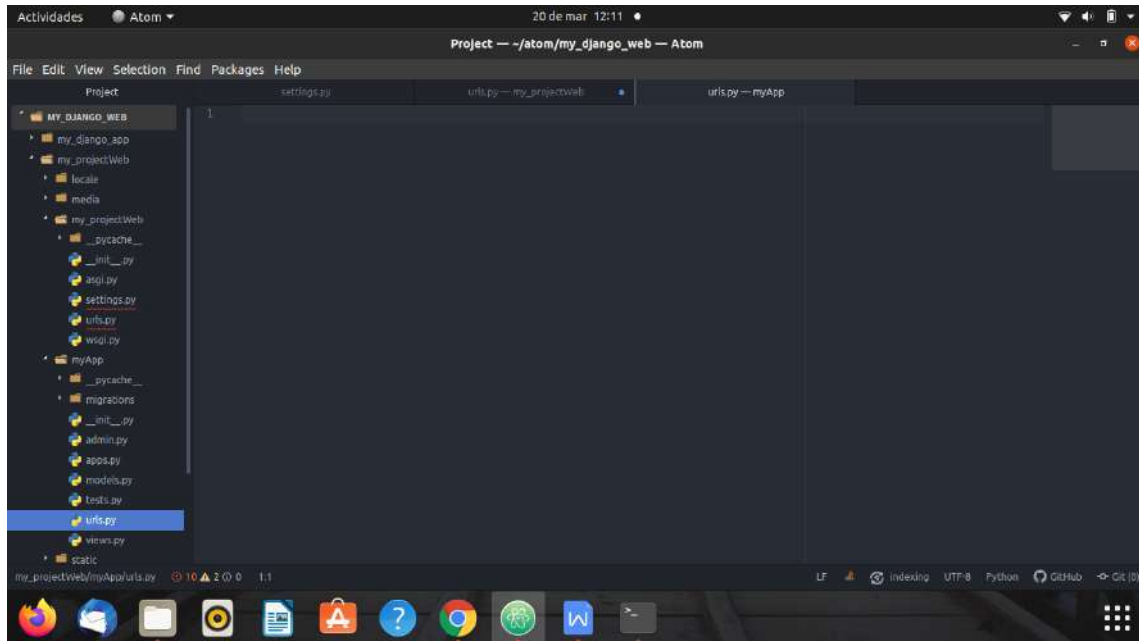


Figura 2.29: Ejemplo de crear el archivo urls.py

Ponemos:

```
from django.conf.urls import url, include  
  
urlpatterns = []
```

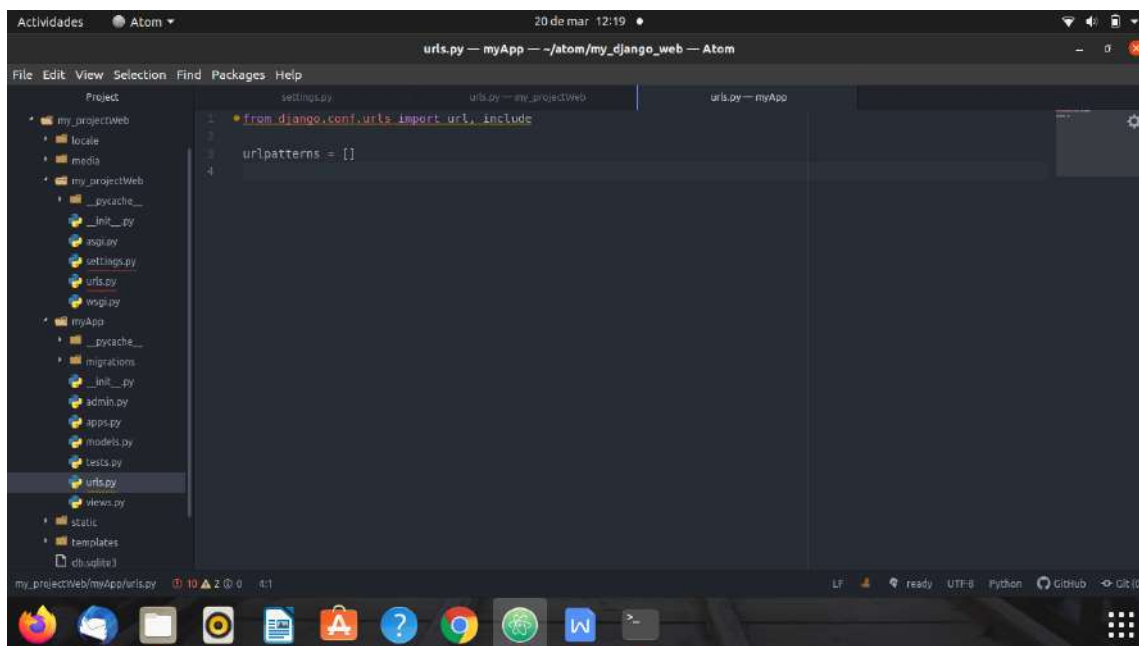


Figura 2.30: Ejemplo de archivo urls.py

Añadimos también a la carpeta del my_projectWeb-->urls.py:

Incluir la aplicación al proyecto

```
from django.conf.urls import url,include
```

Convocar a las urls de las settings

```
from django.conf import settings
```

```
from django.conf.urls.static import static
```

```
urlpatterns = [
```

```
    path('admin/', admin.site.urls),
```

```
    url(r"^", include("myApp.urls")), --> nombramos las urls de la aplicación
```

```
] +static(settings.STATIC_URL,
```

```
document_root=settings.STATICFILES_DIRS)+static(settings.MEDIA_URL,
```

```
document_root=settings.MEDIA_ROOT)
```

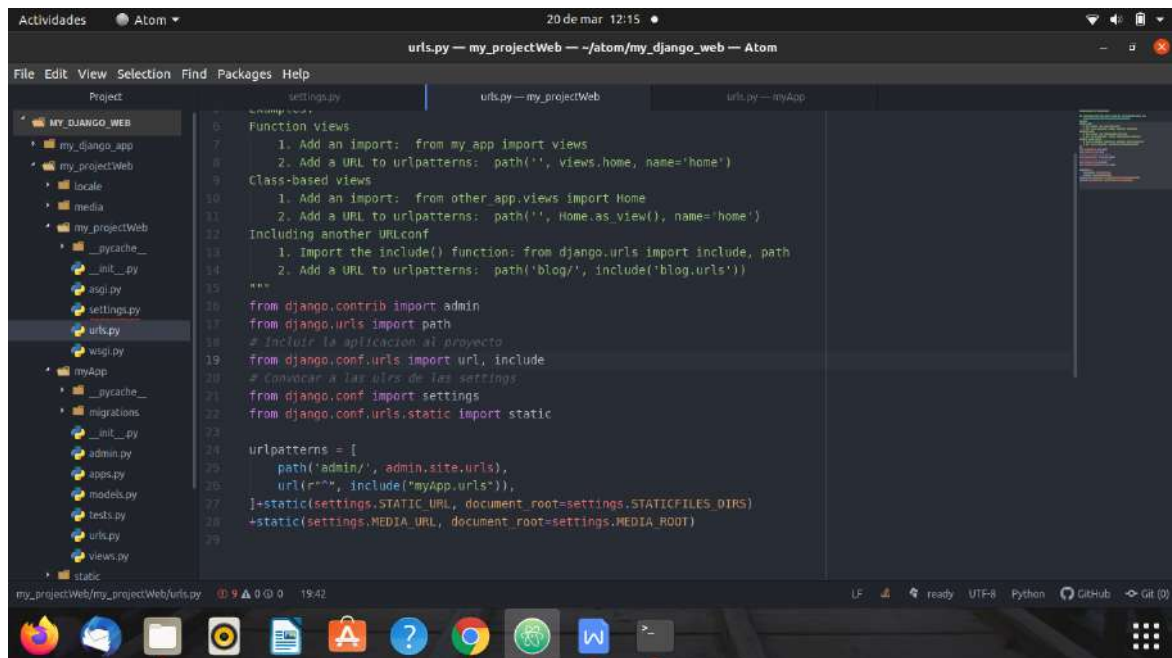


Figura 2.31: Ejemplo de archivo urls.py

2.5 Migración de la aplicación

Cuando ejecutamos la aplicación:

```
python manage.py runserver
```

Se observa un error en color rojo, esto es debido a que nos avisa que necesitamos migrar la aplicación para que funcione correctamente:

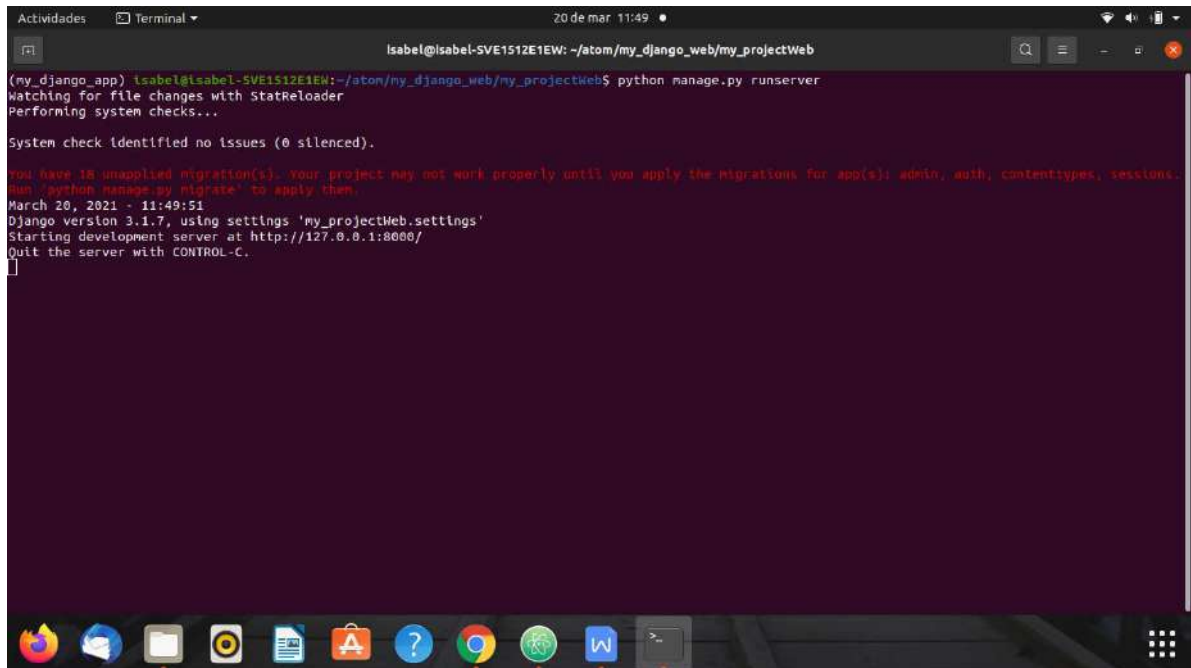
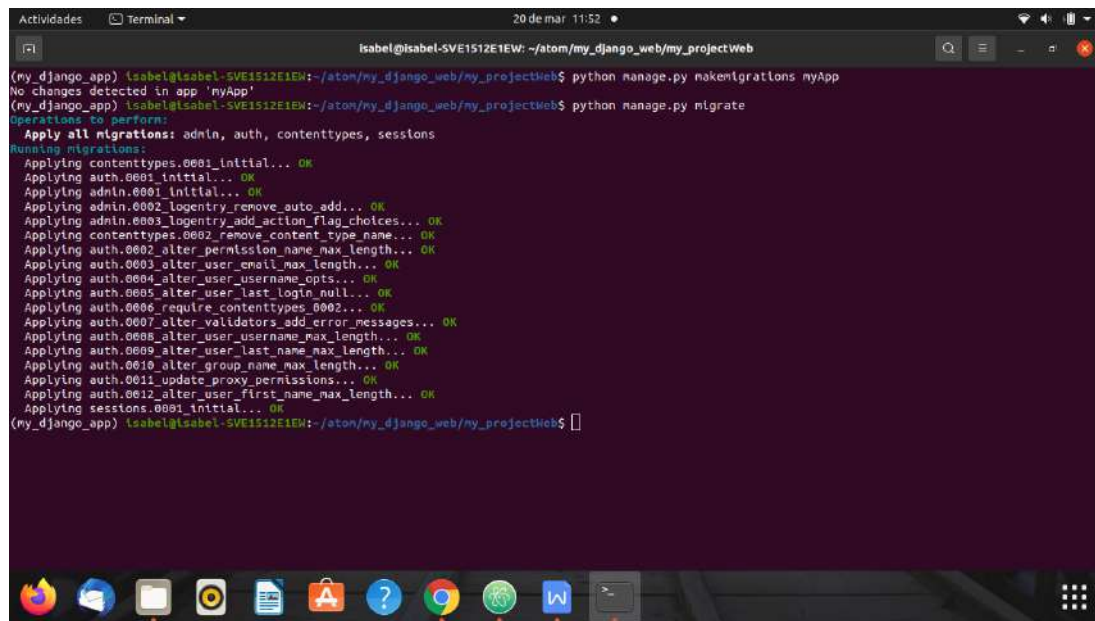


Figura 2.32: Ejemplo ejecutar el proyecto se observa un error de color rojo al arrancarlo.

Para ello será necesario migrar la aplicación mediante dos instrucciones:

```
python manage.py makemigrations myApp  
  
python manage.py migrate
```



```

Actividades Terminal 20 de mar 11:52
isabel@isabel-SVE1512E1EW: ~/atom/my_django_web/my_projectWeb

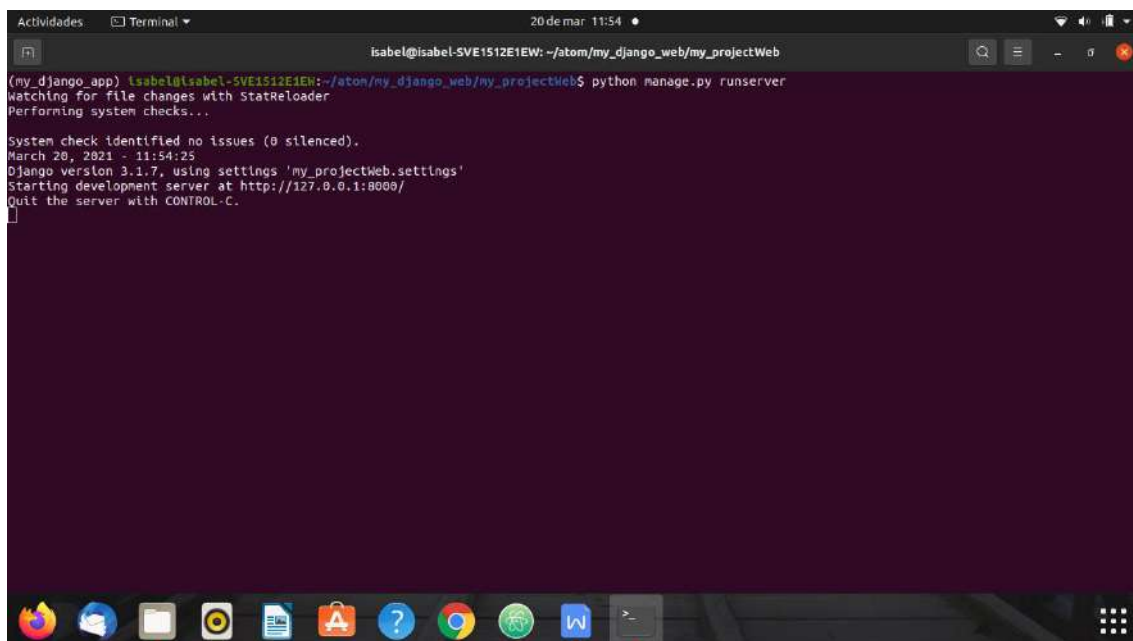
(my_django_app) isabel@isabel-SVE1512E1EW:~/atom/my_django_web/my_projectWeb$ python manage.py makemigrations myApp
No changes detected in app 'myApp'
(my_django_app) isabel@isabel-SVE1512E1EW:~/atom/my_django_web/my_projectWeb$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
(my_django_app) isabel@isabel-SVE1512E1EW:~/atom/my_django_web/my_projectWeb$
  
```

Figura 2.33: Ejemplo ejecutar migrar el modelo de la aplicación

Observamos que se ha migrado con éxito esto nos sirve cuando tengamos un modelo de datos será necesario migrarlo para que los cambios se hayan efectuado con éxito y no nos de errores posteriores.

Ejecutamos de nuevo la aplicación:

```
python manage.py runserver
```



```

Actividades Terminal 20 de mar 11:54
isabel@isabel-SVE1512E1EW: ~/atom/my_django_web/my_projectWeb

(my_django_app) isabel@isabel-SVE1512E1EW:~/atom/my_django_web/my_projectWeb$ python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
March 20, 2021 - 11:54:25
Django version 3.1.7, using settings 'my_projectWeb.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
  
```

Figura 2.34: Ejemplo de ejecutar el proyecto una vez realizada la migración

Observamos que no ha desaparecido el error.

Añadimos estas dos instrucciones al archivo README.md:

- 7) migrar la base de datos: `python manage.py makemigrations myApp`
- 8) `python manage.py migrate`

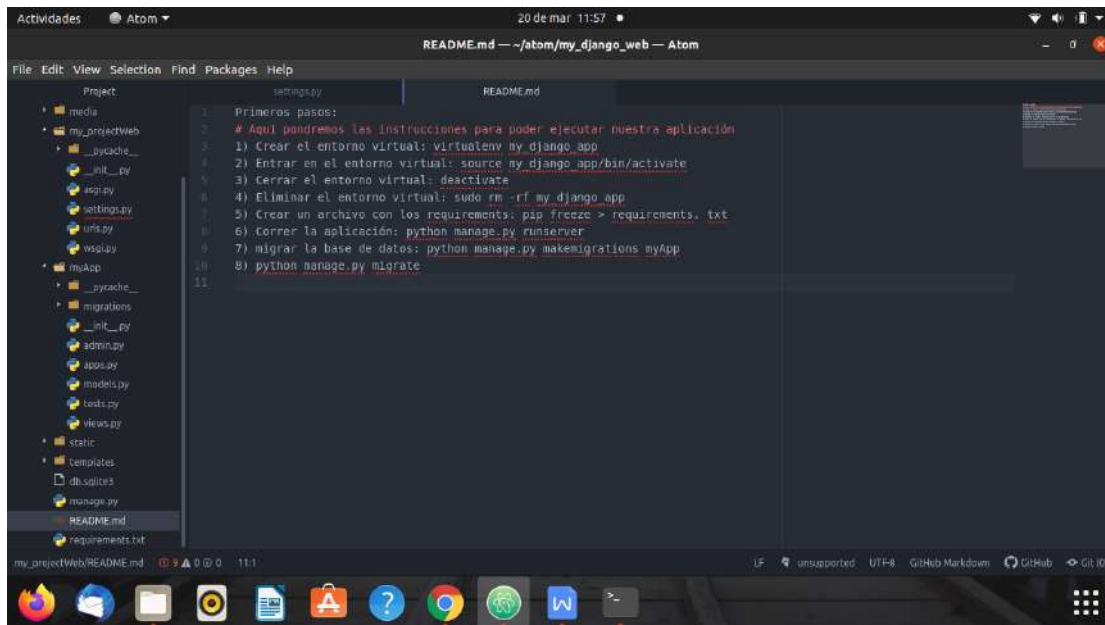


Figura 2.35: Archivo README.md

Estructura de nuestra app:

```
my_django_web/
|-- my_projectWeb/
|   |-- locale/
|   |-- media/
|   |-- myApp/
|       |-- migrations/
|       |   +-- __init__.py
|       |-- __init__.py
|       |-- admin.py
|       |-- apps.py
|       |-- models.py
|       |-- urls.py
|       |-- tests.py
|       +-- views.py
|   |-- my_projectWeb/
|       |-- __init__.py
|       |-- settings.py
|       |-- urls.py
|       |-- wsgi.py
|   |-- static/
|   |-- templates/
|   +-- README.md
|   +-- requirements.txt
+-- my_django_app/
```

carpeta para recursos de idiomas
carpeta para recursos media
archivo donde definiremos nuestras urls
carpeta para recursos CSS e imágenes
carpeta para recursos HTML

Si miramos en el navegador veremos que nos da un error con todos estos cambios, NO TE PREOCUPES SON NORMALES!!!

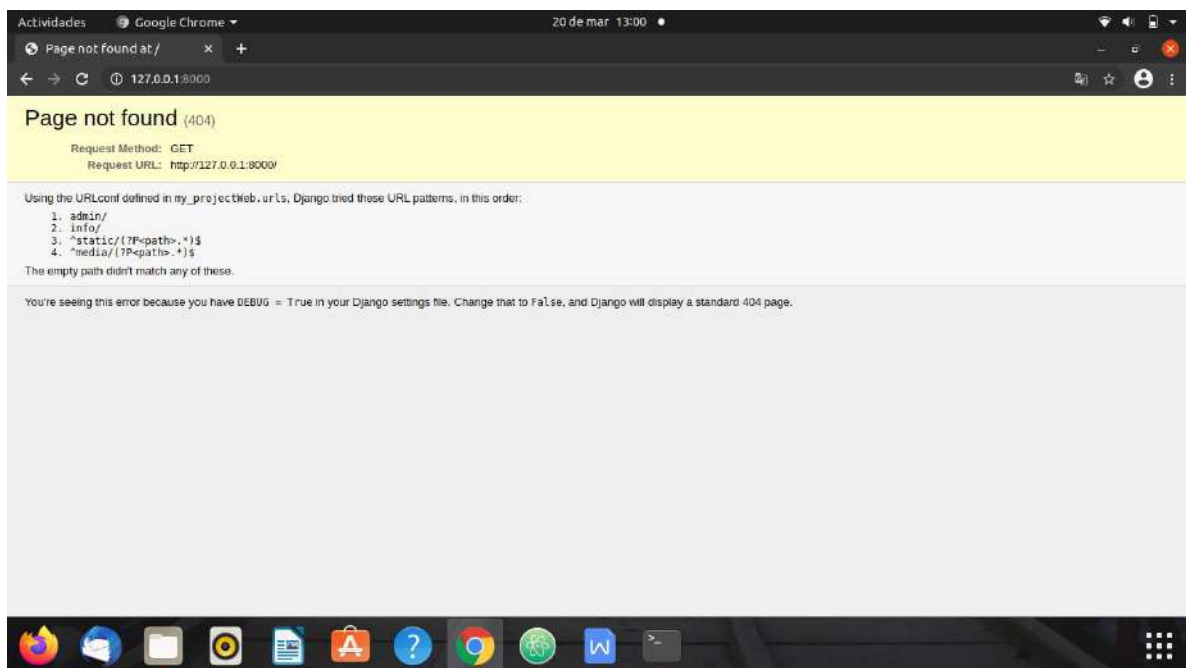



Figura 2.36: Ejemplo de la ejecución de la aplicación en el navegador



Recuerda

No olvides migrar la aplicación cuando realices un cambio en el modelo de datos.

3. PUNTOS CLAVE

- | Crea un Entorno Virtual siempre que empieces un proyecto.
- | La configuración de la aplicación se encuentra en settings.py
- | Migra un proyecto siempre que hagas modificaciones en el modelo de datos.

